

# **Operation Analytics and Investigating Metric Spike**

## **Project Description**

### **Operation Analytics**

Operational Analytics is a category of business analytics that's focused on syncing data from your warehouse directly to your business tools – thus ensuring that everyone across your organization has access to the same data regardless of their technical skills. Operations analytics is a subfield of data analytics that focuses on using data to optimize and improve operational processes within an organization

### **Metric Spike**

A metric spike refers to a sudden and significant increase in a particular metric or measurement that is being tracked or analyzed. This can occur in a wide range of data sets and can be observed over a relatively short period of time.

### **The given project consists of 2 case studies:-**

First is regarding Operation Analytics where job data is provided and number of jobs reviewed, 7 day rolling average of throughput, percentage share of language used and duplicates are found out.

Second is Investigating Metric Spike where user engagement, user growth, weekly retention, weekly engagement and email engagement is determined.

## **Approach**

- Download all Data provided
- Create database with the dataset provided for Case Study 1 (Job Data)
- Upload datasets of Case Study 2 (Investigating metric spike) to MySQL Workbench
- Gather insights of database

- Write queries to find answers
- Analyze the information and make proper decision

## Tech Stack Used

**MySQL Workbench** :- The tech stack used included MySQL Workbench v8.0.39, which was an excellent tool for querying the database, thanks to its ease of access, simple setup, and GUI, as well as its troubleshooting support.

**MS Excel** :- MS Excel was used in the second case study for better visualization.

## Insights

Here's a summary of insights and knowledge gained from an operation analytics and spike metrics investigation project, tailored for company improvement:

- **Performance Trends:** Analyzing spikes revealed performance patterns and trends, allowing the company to understand peak usage times and resource demands better. This insight aids in optimizing resource allocation and planning.
- **Impact Evaluation:** Assessing the impact of spikes on system performance and business operations highlighted critical areas where performance improvements are needed. Understanding the business impact helps prioritize which issues to address first.
- **Enhanced Response Strategies:** Insights led to the development of more effective response strategies, including better monitoring practices, automated alerts, and incident response plans. This improves the company's ability to handle and resolve issues swiftly.

- **Continuous Improvement:** The iterative analysis fostered a culture of continuous improvement, emphasizing the importance of ongoing monitoring and adjustment. This approach enhances overall system resilience and operational efficiency.

Overall, the project provided valuable insights that drive proactive management of system performance, enhance operational stability, and improve the company's ability to respond to and prevent issues related to metric spikes.

### The Observations made from the data are:

#### Case Study 1: Job Data Analysis

You will be working with a table named `job_data` with the following columns:

- **job\_id:** Unique identifier of jobs
- **actor\_id:** Unique identifier of actor
- **event:** The type of event (decision/skip/transfer).
- **language:** The Language of the content
- **time\_spent:** Time spent to review the job in seconds.
- **org:** The Organization of the actor
- **ds:** The date in the format yyyy/mm/dd (stored as text).

#### Query:

```

• create database project3;
• show databases;
• use project3;

# case study 1 : job data analysis

• CREATE TABLE job_data
  (ds DATE,
   job_id INT NOT NULL,
   actor_id INT NOT NULL,
   event VARCHAR(15) NOT NULL,
   language VARCHAR(15) NOT NULL,
   time_spent INT NOT NULL,
   org CHAR(2)
  );

```

```



INSERT INTO job_data (ds, job_id, actor_id, event, language, time_spent, org)
VALUES ('2020-11-30', 21, 1001, 'skip', 'English', 15, 'A'),
      ('2020-11-30', 22, 1006, 'transfer', 'Arabic', 25, 'B'),
      ('2020-11-29', 23, 1003, 'decision', 'Persian', 20, 'C'),
      ('2020-11-28', 23, 1005, 'transfer', 'Persian', 22, 'D'),
      ('2020-11-28', 25, 1002, 'decision', 'Hindi', 11, 'B'),
      ('2020-11-27', 11, 1007, 'decision', 'French', 104, 'D'),
      ('2020-11-26', 23, 1004, 'skip', 'Persian', 56, 'A'),
      ('2020-11-25', 20, 1003, 'transfer', 'Italian', 45, 'C');

Select * from job_data;


```

## Output:


Result Grid



Filter Rows:

Export:



Wrap Cell Content:



	ds	job_id	actor_id	event	language	time_spent	org
▶	2020-11-30	21	1001	skip	English	15	A
	2020-11-30	22	1006	transfer	Arabic	25	B
	2020-11-29	23	1003	decision	Persian	20	C
	2020-11-28	23	1005	transfer	Persian	22	D
	2020-11-28	25	1002	decision	Hindi	11	B
	2020-11-27	11	1007	decision	French	104	D
	2020-11-26	23	1004	skip	Persian	56	A
	2020-11-25	20	1003	transfer	Italian	45	C

## Tasks:

### 1. Jobs Reviewed Over Time:

Objective: Calculate the number of jobs reviewed per hour for each day in November 2020.

Task: Write an SQL query to calculate the number of jobs reviewed per hour for each day in November 2020.

## Query:

#Task A jobs reviewed over time

```
SELECT avg(t) as 'avg jobs reviewed per day per hour',  
avg(p) as 'avg jobs reviewed per day per second'  
from  
> (select  
ds,  
((count(job_id)*3600)/sum(time_spent)) as t,  
((count(job_id))/sum(time_spent)) as p  
from  
job_data  
where  
month(ds)=11  
- group by ds) a;
```

Output:

Result Grid		Filter Rows:	Export:	Write
	avg jobs reviewed per day per hour	avg jobs reviewed per day per second		
▶	126.18048333	0.03505000		

## 2. Throughput Analysis:

Objective: Calculate the 7-day rolling average of throughput (number of events per second).

Task: Write an SQL query to calculate the 7-day rolling average of throughput.

Query:

#Task B Throughput Analysis

```
SELECT ROUND(COUNT(event)/SUM(time_spent), 2) AS 'Weekly Throughput' FROM job_data;
```

```
SELECT ds as Dates, ROUND(COUNT(event)/SUM(time_spent),2) AS 'Daily Throughput' FROM job_data  
GROUP BY ds ORDER BY ds;
```

## Output:

Result Grid	Filter Rows:
Dates	Daily Throughput
2020-11-25	0.02
2020-11-26	0.02
2020-11-27	0.01
2020-11-28	0.06
2020-11-29	0.05
2020-11-30	0.05

### 3. Language Share Analysis:

Objective: Calculate the percentage share of each language in the last 30 days.




Task: Write an SQL query to calculate the percentage share of each language over the last 30 days.

## Query:

```
#Task C Language share analysis
```

```
select language as Languages, ROUND(100 * COUNT(*)/total, 2) as Percentage, sub.total
from job_data
cross join (select COUNT(*) AS total from job_data) as sub
group by language, sub.total;
```

## Output:

Result Grid			 Filter Rows:
	Languages	Percentage	total
	English	12.50	8
	Arabic	12.50	8
	Persian	37.50	8
	Hindi	12.50	8
	French	12.50	8
	Italian	12.50	8

#### 4. Duplicate Rows Detection:

Objective: Identify duplicate rows in the data.

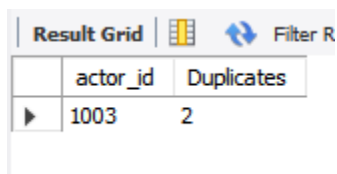
Task: Write an SQL query to display duplicate rows from the job\_data table.

Query:

```
#Task D Duplicate rows detection
```

```
select actor_id, count(*) as Duplicates from job_data  
group by actor_id having count(*) > 1;
```

Output:



	actor_id	Duplicates
▶	1003	2

#### Case Study 2: Investigating Metric Spike

we will be working with three tables:

- **users:** Contains one row per user, with descriptive information about that user's account.

Query:

```
# table 1 users

create table users(
  user_id int,
  created_at varchar(100),
  company_id int,
  language varchar(50),
  activated_at varchar(100),
  state varchar(50));

SHOW VARIABLES LIKE 'secure_file_priv';

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users.csv"
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

select * from users;

alter table users add column temp_created_at datetime;

update users set temp_created_at = STR_TO_DATE(created_at, '%d-%m-%Y %H:%i');

alter table users DROP COLUMN created_at;

alter table users change column temp_created_at created_at datetime;
```

**Output:**



Result Grid						
Filter Rows:						
Export:						
Wrap Cell Content:						
	user_id	company_id	language	activated_at	state	created_at
▶	0	5737	english	01-01-2013 21:01	active	2013-01-01 20:59:00
	3	2800	german	01-01-2013 18:42	active	2013-01-01 18:40:00
	4	5110	indian	01-01-2013 14:39	active	2013-01-01 14:37:00
	6	11699	english	01-01-2013 18:38	active	2013-01-01 18:37:00
	7	4765	french	01-01-2013 16:20	active	2013-01-01 16:19:00
	8	2698	french	01-01-2013 04:40	active	2013-01-01 04:38:00
	11	3745	english	01-01-2013 08:09	active	2013-01-01 08:07:00
	13	4025	english	02-01-2013 12:29	active	2013-01-02 12:27:00
	15	4259	english	02-01-2013 15:41	active	2013-01-02 15:39:00
	17	5025	japanese	02-01-2013 10:57	active	2013-01-02 10:56:00

- **events:** Contains one row per event, where an event is an action that a user has taken (e.g., login, messaging, search).

### Query:

```
#table 2 events
create table events (
  user_id int,
  occurred_at varchar(100),
  event_type varchar(50),
  event_name varchar(100),
  location varchar(50),
  device varchar(50),
  user_type int
);
```

```
SHOW VARIABLES LIKE 'secure_file_priv';
```

```
LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/events.csv"
INTO TABLE events
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

```
select * from events;
```

```
alter table events add column temp_occurred_at datetime;
```

```
update events set temp_occurred_at = STR_TO_DATE(occurred_at, '%d-%m-%Y %H:%i');
```

```
alter table events drop column occurred_at;
```

```
alter table events change column temp_occurred_at occurred_at datetime;
```

## Output:

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Fetch rows:

	user_id	event_type	event_name	location	device	user_type	occurred_at
▶	10522	engagement	login	Japan	dell inspiron notebook	3	2014-05-02 11:02:00
	10522	engagement	home_page	Japan	dell inspiron notebook	3	2014-05-02 11:02:00
	10522	engagement	like_message	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
	10522	engagement	view_inbox	Japan	dell inspiron notebook	3	2014-05-02 11:04:00
	10522	engagement	search_run	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
	10522	engagement	search_run	Japan	dell inspiron notebook	3	2014-05-02 11:03:00
	10612	engagement	login	Netherlands	iphone 5	1	2014-05-01 09:59:00
	10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:00:00
	10612	engagement	send_message	Netherlands	iphone 5	1	2014-05-01 10:00:00
	10612	engagement	home_page	Netherlands	iphone 5	1	2014-05-01 10:01:00
	10612	engagement	like_message	Netherlands	iphone 5	1	2014-05-01 10:01:00
	10612	engagement	home_page	Netherlands	iphone 5	1	2014-05-01 10:02:00
	10612	engagement	view_inbox	Netherlands	iphone 5	1	2014-05-01 10:02:00

- **email\_events**: Contains events specific to the sending of emails.

## Query:

```

#table 3 email events
) create table emailEvents(
  user_id int,
  occurred_at varchar(100),
  action varchar(100),
  user_type int
);


LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/email_events.csv"
INTO TABLE emailEvents
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;


select * from emailEvents;

```

## Output:

Result Grid

 Filter Rows:

Export: 

Wr

	user_id	occurred_at	action	user_type
▶	0	06-05-2014 09:30	sent_weekly_digest	1
	0	13-05-2014 09:30	sent_weekly_digest	1
	0	20-05-2014 09:30	sent_weekly_digest	1
	0	27-05-2014 09:30	sent_weekly_digest	1
	0	03-06-2014 09:30	sent_weekly_digest	1
	0	03-06-2014 09:30	email_open	1
	0	10-06-2014 09:30	sent_weekly_digest	1
	0	10-06-2014 09:30	email_open	1
	0	17-06-2014 09:30	sent_weekly_digest	1
	0	17-06-2014 09:30	email_open	1
	0	24-06-2014 09:30	sent_weekly_digest	1
	0	01-07-2014 09:30	sent_weekly_digest	1
	0	08-07-2014 09:30	sent_weekly_digest	1
	0	15-07-2014 09:30	sent_weekly_digest	1

emailEvents 8 ×

## Tasks:

### 1. Weekly User Engagement:

Objective: Measure the activeness of users on a weekly basis.

Task: Write an SQL query to calculate the weekly user engagement.

Query:

```
#Task A weekly user engagement
select extract(week from occurred_at) as week_number,
count(distinct user_id) as active_user
from events
where event_type='engagement'
group by week_number;
```

Output:

Result Grid			Filter Rows:
	week_number	active_user	
▶	17	663	
	18	1068	
	19	1113	
	20	1154	
	21	1121	
	22	1186	
	23	1232	
	24	1275	
	25	1264	
	26	1302	
	27	1372	
	28	1365	
	29	1376	
	30	1467	
	31	1299	
	32	1225	
	33	1225	
	34	1204	
	35	104	

## 2. User Growth Analysis:

Objective: Analyze the growth of users over time for a product.

Task: Write an SQL query to calculate the user growth for the product.

### Query:

#Task B User growth analysis

```
SELECT YEAR, WEEK_NUMBER, NUM_USERS, SUM(NUM_USERS) OVER(ORDER BY YEAR, WEEK_NUMBER ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
AS CUMULATIVE_ACTIVE_USERS
FROM ( SELECT EXTRACT(YEAR FROM ACTIVATED_AT ) AS YEAR,
EXTRACT(WEEK FROM ACTIVATED_AT) AS WEEK_NUMBER, COUNT(DISTINCT USER_ID) AS NUM_USERS FROM users
WHERE STATE='ACTIVE' GROUP BY YEAR, WEEK_NUMBER ORDER BY YEAR, WEEK_NUMBER)A;
```

### Output:

YEAR	WEEK_NUMBER	NUM_USERS	CUMULATIVE_ACTIVE_USERS
2013	0	23	23
2013	1	30	53
2013	2	48	101
2013	3	36	137
2013	4	30	167
2013	5	48	215
2013	6	38	253
2013	7	42	295
2013	8	34	329
2013	9	43	372
2013	10	32	404
2013	11	31	435
2013	12	33	468
2013	13	39	507
2013	14	35	542
2013	15	43	585
2013	16	46	631
2013	17	49	680
2013	18	44	724
2013	19	57	781
2013	20	39	820

2013	21	49	869
2013	22	54	923
2013	23	50	973
2013	24	45	1018
2013	25	57	1075
2013	26	56	1131
2013	27	52	1183
2013	28	72	1255
2013	29	67	1322
2013	30	67	1389
2013	31	67	1456
2013	32	71	1527
2013	33	73	1600
2013	34	78	1678
2013	35	63	1741
2013	36	72	1813
2013	37	85	1898
2013	38	90	1988

2013	39	84	2072
2013	40	87	2159
2013	41	73	2232
2013	42	99	2331
2013	43	89	2420
2013	44	96	2516
2013	45	91	2607
2013	46	88	2695
2013	47	102	2797
2013	48	97	2894
2013	49	116	3010
2013	50	124	3134
2013	51	102	3236
2013	52	47	3283
2014	0	83	3366
2014	1	126	3492
2014	2	109	3601
2014	3	113	3714
2014	4	130	3844
2014	5	133	3977
2014	6	135	4112
2014	7	125	4237
2014	8	129	4366
2014	9	133	4499

[Activate Windows](#)  
[Go to Settings to activate Windows.](#)

2014	9	133	4499
2014	10	154	4653
2014	11	130	4783
2014	12	148	4931
2014	13	167	5089
2014	14	162	5260
2014	15	164	5424
2014	16	179	5603
2014	17	170	5773
2014	18	163	5936
2014	19	185	6121
2014	20	176	6297
2014	21	183	6480
2014	22	196	6676
2014	23	196	6872
2014	24	229	7101
2014	25	207	7308
2014	26	201	7509
2014	27	222	7731
2014	28	215	7946
2014	29	221	8167
2014	30	238	8405

2014	31	193	8598
2014	32	245	8843
2014	33	261	9104
2014	34	259	9363
2014	35	18	9381

### 3. Weekly Retention Analysis:

Objective: Analyze the retention of users on a weekly basis after signing up for a product.

Task: Write an SQL query to calculate the weekly retention of users based on their sign-up cohort.

**Query:**

```

#Task C weekly retention analysis
select count(user_id) total_engaged_users,
       sum(case when retention_week = 1 then 1 else 0 end) as retained_users
from
> (
  select a.user_id,
         a.sign_up_week,
         b.engagement_week,
         b.engagement_week - a.sign_up_week as retention_week
  from
  > (
  > (select distinct user_id, extract(week from occurred_at) as sign_up_week
  from events
  where event_type = 'signup_flow'
  and event_name = 'complete_signup'
  - and extract(week from occurred_at)=18)a
  left join
  > (select distinct user_id, extract(week from occurred_at) as engagement_week
  from events
  - where event_type = 'engagement')b
  on a.user_id = b.user_id
  - )
  group by user_id
  order by user_id

```

---

Output:

total_engaged_users	retained_users
317	236

#### 4. Weekly Engagement Per Device:

Objective: Measure the activeness of users on a weekly basis per device.

Task: Write an SQL query to calculate the weekly engagement per device.


Query:



```
#Task D weekly engagement per device
select extract(year from occurred_at) as year, extract(week from occurred_at) as week,
device, count(distinct user_id) as count
from events
where event_type = 'engagement'
group by 1,2,3
order by 1,2,3;
```

## Output:

Result Grid



Filter Rows:

	year	week	device	count
▶	2014	17	acer aspire desktop	9
	2014	17	acer aspire notebook	20
	2014	17	amazon fire phone	4
	2014	17	asus chromebook	21
	2014	17	dell inspiron desktop	18
	2014	17	dell inspiron notebook	46
	2014	17	hp pavilion desktop	14
	2014	17	htc one	16
	2014	17	ipad air	27
	2014	17	ipad mini	19
	2014	17	iphone 4s	21
	2014	17	iphone 5	65
	2014	17	iphone 5s	42
	2014	17	kindle fire	6

Result 13

×

year	week	device	count
2014	17	kindle fire	6
2014	17	lenovo thinkpad	86
2014	17	mac mini	6
2014	17	macbook air	54
2014	17	macbook pro	143
2014	17	nexus 10	16
2014	17	nexus 5	40
2014	17	nexus 7	18
2014	17	nokia lumia 635	17
2014	17	samsung galaxy tablet	8
2014	17	samsung galaxy note	7
2014	17	samsung galaxy s4	52
2014	17	windows surface	10
2014	18	acer aspire desktop	26

year	week	device	count
2014	18	acer aspire notebook	33
2014	18	amazon fire phone	9
2014	18	asus chromebook	42
2014	18	dell inspiron desktop	58
2014	18	dell inspiron notebook	77
2014	18	hp pavilion desktop	37
2014	18	htc one	19
2014	18	ipad air	52
2014	18	ipad mini	30
2014	18	iphone 4s	46
2014	18	iphone 5	113
2014	18	iphone 5s	73
2014	18	kindle fire	27
2014	18	lenovo thinkpad	153

Result 13 ×

## 5. Email Engagement Analysis:

Objective: Analyze how users are engaging with the email service.

Task: Write an SQL query to calculate the email engagement metrics.

### Query:

```
#Task E email engagement analysis
select
100 * sum(case when email_cat = 'email_open' then 1 else 0 end)/sum(case when email_cat = 'email sent' then 1 else 0 end) as email_open_rate,
100 * sum(case when email_cat = 'email clicked' then 1 else 0 end)/sum(case when email_cat = 'email sent' then 1 else 0 end) as email_click_rate
from (select*,
Case
When action in ('sent_weekly_digest','sent_reengagement_email') then 'email_sent'
when action in('email_open') then 'email_open'
when action in('email_clickthrough') then 'email_clicked'
end as email_cat
from emailEvents) sub;
```

### Output:

	email_open_rate	email_click_rate
▶	31.1921	10.4745

## Result:

Through this project, I was able to understand how important **Operational Analytics** is for an organizations as it helps in identifying and understanding areas where **improvement** is required.

In this project I was able to get insights about various questions like rate of job reviews, share of languages, patterns of user engagement on weekly basis, growth of users etc. which can be **communicated** to the management team as per the requirements using which they can make proper **data-driven decisions**.